Concealing Cyber-Decoys using Two-Sided Feature Deception Games

Mohammad Sujan Miah
The University of Texas at El Paso
msmiah@miners.utep.edu

Marcus Gutierrez
The University of Texas at El Paso
mgutierrez22@miners.utep.edu

Oscar Veliz
The University of Texas at El Paso
osveliz@utep.edu

Omkar Thakoor
University of Southern California
othakoor@usc.edu

Christopher Kiekintveld
The University of Texas at El Paso
cdkiekintveld@utep.edu

Abstract

An increasingly important tool for securing computer networks is the use of deceptive decoy objects (e.g., fake hosts, accounts, or files) to detect, confuse, and distract attackers. One of the well-known challenges in using decoys is that it can be challenging to design effective decoys that are hard to distinguish from real objects, especially against sophisticated attackers who may be aware of the use of decoys. A key issue is that both real and decoy objects have observable features that may give the attacker the ability to distinguish one from the other. However, a defender deploying decoys may be able to modify some features of either the real or decoy objects (at some cost) making the decoys more effective. We present a game-theoretic model of two-sided deception that models this scenario. We present an empirical analysis of this model to show strategies for effectively concealing decoys, as well as some limitations of decoys for cybersecurity.

1. Introduction

Both civilian and military computer networks are under increasing threat from cyber attacks, with the most significant threat posed by Advanced Persistent Threat (APT) actors. These attackers use sophisticated methods to compromise networks and remain inside, establishing greater control and staying for long periods to gather valuable data and intelligence. These attackers seek to remain undetected, and estimates from APT attacks show that they are often present in a network for months before they are detected [1].

Cyber deception methods use deceptive decoy objects like fake hosts (honeypots), network traffic, files, and even user accounts to counter attackers in a variety of ways [2, 3, 4]. They can create confusion for attackers, make them more hesitant and less effective in executing further attacks, and can help to gather information about the behavior and tools of various attackers. They can also increase the ability of defenders to detect malicious activity and actors in the network. This deception is especially critical in the case of APT attackers, who are often cautious and skilled at evading detection [5]. Widespread and effective use of honeypots and other deceptive objects is a promising approach for combating this class of attackers.

However, the effectiveness of honeypots and other deceptive objects depends crucially on whether the honeypot creators can design them to look similar enough to real objects, to prevent honeypot detection and avoidance. This design goal especially holds for APT threats, which are likely to be aware of the use of such deception technologies and will actively seek to identify and avoid honeypots, and other deceptive objects, in their reconnaissance [5, 6]. A well-known problem with designing successful honeypots is that they often have characteristics that can be observed by an attacker that will reveal the deception [7]. Examples of such characteristics include the patterns of network traffic to a honeypot, the response times to queries, or the configuration of services which are not similar to real hosts in the network. However, with some additional effort, these characteristics can be made more effective in deception (e.g., by simulating more realistic traffic to and from honeypots).

We introduce a game-theoretic model of the problem of designing effective decoy objects that can fool even a sophisticated attacker. In our model, real and fake objects may naturally have different distributions of characteristic features than an attacker could use to tell them apart. However, the defender can make some (costly) modifications to *either* the real or the fake objects to make them harder to distinguish. This model captures some key aspects of cyber deception that are missing from other game-theoretic models. In particular, we focus on whether the defender can design convincing decoy objects, and what the limitations of

URI: https://hdl.handle.net/10125/63974
978-0-9981331-3-3
(CC BY-NC-ND 4.0)

deception are if some discriminating features of real and fake objects are not easily maskable.

We present several analyses of fundamental questions in cyber deception based on our model. We analyze how to measure the informativeness of the signals in our model, and then consider how effectively the defender can modify the features to improve the effectiveness of deception in various settings. We show how different variations in the costs of modifying the features can have a significant impact on the effects of deception. We also consider the differences between modifying only the features of deceptive objects and being able to modify both real and deceptive objects (two-sided deception). While this is not always necessary, in some cases, it is essential to enable effective deception. We also consider deception against naïve attackers, and how this compares to the case of sophisticated attackers. Finally, we discuss how our model relates to work in adversarial learning and how this model could be applied beyond the case of honeypots to, for example, generating decoy network traffic.

2. Motivating Domain and Related Work

While the model we present may apply to many different types of deception and deceptive objects, we will focus on honeypots as a specific case to make our discussion more concrete and give an example of how this model captures essential features of real-world deception problems. Honeypots have had a considerable impact on cyber defense in the 30 years since they were first introduced [8].

Over time, honeypots have been used for many different purposes, and have evolved to more sophisticated designs with more advanced abilities to mimic real hosts and to capture useful information about attackers [9, 10, 11]. The sophistication of honeypots can vary dramatically, from limited low-interaction honeypots to sophisticated high-interaction honeypots [9, 12, 13].

Here, we do not focus on the technological advancements of honeypots, but rather on the game-theoretic investigation of honeypot deception. There have been numerous works that emphasize this game-theoretic approach to cyber deception as well. Our work builds upon the Honeypot Selection Game (HSG), described by Píbil et al. [14, 3]. Much like the HSG, we model the game using an extensive form game. We extend the HSG model with the introduction of *features*, which are modifiable tokens in each host that enable more robust deceptions and allow to model more realistic settings. Several game-theoretic models have been established for other cyber defense problems [15, 16, 17, 18], specifically for deception as well [19, 20], however these consider attribute obfuscation as the means of deception rather than use of decoy objects.

[21] notably investigates the use of honeypots in the smart grid to mitigate denial-of-service attacks through the lens of Bayesian games. [22] also model honeypots mitigating denial-of-service attacks in a similar fashion but in the Internet-of-Things domain. [23] tackles a similar "honeypots to protect social networks against DDoS attacks" problem with Bayesian game modeling. These works demonstrate the broad domains where honeypots can aid. This work differs in that we do not model a Bayesian incomplete information game.

A couple of works also consider the notion of two-sided deception, where the defender deploys not only *real*-looking honeypots but also *fake*-looking real hosts. Rowe et al. demonstrate that using two-sided deception offers an improved defense by scaring off attackers [24]. Caroll and Grosu introduced the signaling deception game where signals bolster a deployed honeypot's deception [25]. Our work differs in that we define specific features (signals) that can be altered and revealed to the attacker. Shi et al. introduce the mimicry honeypot framework, which combines real nodes, honeypots, and *fake*-looking honeypots to derive equilibria strategies to bolster defenses [26]. They validated their work in a simulated network. This notion of two-sided deception is quickly becoming a reality; De Gaspari et al. provided a prototype proof-of-concept system where production systems also engaged in active deception [27].

3. Honeypot Feature Selection Game

We now present a formal model of the Honeypot Feature Selection Game (HFSG). This game models the optimal decisions for a player (the defender) who is trying to disguise the identity of real and fake objects so that the other player (the attacker) is not able to reliably distinguish between them. Each object in the game is associated with a vector of observable features (characteristics) that provides an informative signal that the attacker can use to detect fake objects more reliably. The defender can make (limited) changes to these observable features, at a cost. Unlike many models of deception, we consider the possibility that the defender can make changes to both the real and fake objects; we refer to this as 2-sided deception.

The original feature vector is modeled as a move by nature in a Bayesian game. Real and fake objects have different probabilities of generating every

possible feature vector. How useful the features are to the attacker depends on how similar the distributions for generating the feature vectors are; very similar distributions have little information while very different distributions may precisely reveal which objects are real or fake. The defender can observe the features and may choose to pay some cost to modify a subset of the features. The attacker observes this modified set of feature vectors and chooses which object to attack. The attacker receives a positive payoff if he selects a real object, and a negative one if he selects a honeypot.

To keep the initial model simple, we focus on binary feature vectors to represent the signals. We will also assume that the defender can modify a maximum of one feature. Both of these can be generalized in a straightforward way, at the cost of a larger and more complex model.

3.1. Formal definition of Honeypot Feature Selection Game

We now define the Honeypot Feature Selection Game (HFSG) formally by the tuple $G = (K^{\mathrm{r}}, K^{\mathrm{h}}, N, v^{\mathrm{r}}, v^{\mathrm{h}}, C^{\mathrm{r}}, C^{\mathrm{h}}, P^{\mathrm{r}}, P^{\mathrm{h}}, \tau, \chi)$.

- K^{r} denotes the set of real hosts and K^{h} denotes the set of honeypots. Altogether, we have the complete set of hosts $K = K^{\mathrm{r}} \cup K^{\mathrm{h}}$. We denote the cardinalities of these by $k = |K|$, $r = |K^{\mathrm{r}}|$, $h = |K^{\mathrm{h}}|$.
- $[n]$ is the set of features that describe any given host. The sequence of feature values of a host is referred to as its *configuration*. Thus, the set of different possible configurations is $\{0,1\}^n$.
- v^{r}, v^{h} denote the importance values of the real hosts and honeypots resp.
- C^{r}, C^{h} denote the cost vectors associated with modifying a single feature of a real host and a honeypot resp., and are indexed by the set of features N. Thus, C^{r}_i is the cost of modifying the i^{th} feature of a real host.
- $P^{\mathrm{r}} : \{0,1\}^n \to [0,1]$ is probability distribution over feature vectors for real hosts
- $P^{\mathrm{h}} : \{0,1\}^n \to [0,1]$ is the probability distribution over feature vectors for honeypots
- The collection of all possible information sets is denoted by τ.
- $\chi : \{0,1\}^{kn} \times D \to \tau$ is a function that given the initial network and a defender action, outputs the attacker's resultant information set $I \in \tau$. Here, D is the set of defender actions.

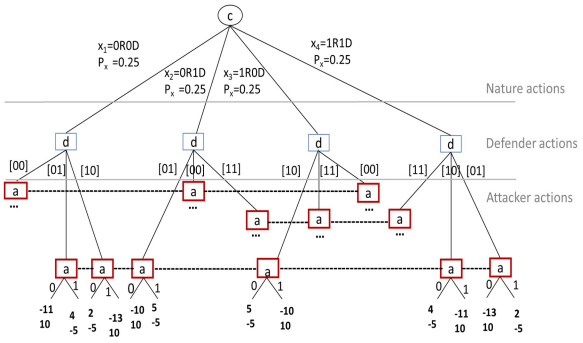


Figure 1. The extensive form game tree with one real host, one honeypot and 1 feature in each host. The importance value of real host is 10 whereas the modification cost of a feature is 3. The same values for the honeypot are 5, 1 resp.

An example of a small HFSG with 1 real host, 1 honeypot, and 1 feature for each host is shown in Figure 1. The probability distributions $P^{\mathrm{r}}(0) = P^{\mathrm{r}}(1) = 0.5$, and $P^{\mathrm{h}}(0) = P^{\mathrm{h}}(1) = 0.5$ are randomly generated for each feature combination.

3.2. Nature Player Actions

We assume that both players know the probability distributions P^r and P^h that define how the feature vectors are selected by nature for real and honeypot hosts, respectively. Nature generates the network configurations as per the distributions P^{r} and P^{h}. Thus, the network state $x = (x_1, \ldots, x_k)$ is generated as per the joint distribution P^{x} where $P^{\mathrm{x}}(x) = \prod_{i=1}^{r} P^{\mathrm{r}}(x_i) \times \prod_{i=r+1}^{k} P^{\mathrm{h}}(x_i)$. Both players can compute the distribution P^{x}. For example, in Figure 1 $P^{\mathrm{x}} = 0.25$ for network 0R1D is calculated from $P^{\mathrm{r}}(0) = 0.5$ and $P^{\mathrm{h}}(1) = 0.5$.

3.3. Defender Actions

The defender observes the network configuration $x \in X$, selected by nature as per probability distribution P^{x}. Then he chooses an appropriate action $d \in D$, which is to change at most one feature of any single host. Thus, D has $nk + 1$ different actions. This action results in a configuration $x' \in \{0,1\}^{nk}$ that the attacker observes, defining his information set $I \in \tau$ as described previously. In the example of Figure 1, given the initial network configuration 0R0D, the defender can alter a feature which results into 0R1D or 1R0D, or make no change leading to 0R0D as the attacker's observation.

3.4. Attacker Actions

The attacker observes the set of feature vectors for each network, but does not directly know which ones are real and which are honeypot. Thus, any permutation of the host configurations is perceived identically by the attacker. Hence, the attacker's information set is merely characterized by the combination of the host configurations and thus represented as a multiset on the set of host configurations as the Universe. For example, in Figure 1, the networks 0R1D and 1R0D belong to the same information set. Given the attacker's information set, he decides which host to attack. When indexing the attack options, we write the information set as an enumeration of the k host configurations, and we assume a lexicographically sorted order as a convention. Given this order, we use a binary variable a_i^I to indicate that when he is in the information set I, the attacker's action is to attack host $i \in K$.

3.5. Utility Functions

A terminal state t in the extensive form game tree is characterized by the sequence of actions that the players (nature, defender, attacker) take. The utilities of the players can be identified based on the terminal state that the game reaches. Thus, given a terminal state t as a tuple (x, j, a) of the player actions, we define a function $U(t) = U(x, j, a)$ such that the attacker gains this value while the defender loses as much. That is, this function serves as the zero-sum component of the player rewards. In particular, if the action a in the information set $\chi(x, j)$ corresponds to a real host, then $U(x, j, a) = v^{\mathrm{r}}$, whereas, if it corresponds to a honeypot, then $U(x, j, a) = -v^{\mathrm{h}}$. Intuitively, the successful identification of a real host gives a positive reward to the attacker otherwise gives a negative reward that is equal to the importance value of a honeypot. The expected rewards are computed by summing over the terminal states and considering the probabilities of reaching them. Finally, the defender additionally also incurs the feature modification cost C_i^{r} or C_j^{h} if his action involved modifying i^{th} feature of a real host or j^{th} feature of a honeypot respectively.

3.6. Defender's Linear Program

We can solve this extensive form game with imperfect information using a linear program. For solving this game in sequence form [28], we create a path from the root node to the terminal node that is a valid sequence and consists of a list of actions for all players. Then we compute defender's behavioral strategies on all valid sequences using a formulated LP as follows, where U_d and U_a are the utilities of the defender and the attacker. To solve the program, we construct a matrix $X[0 : 2^{kn}]$ of all possible network configurations, and then the defender chooses a network $x \in X$ to modify. In network x, any action d of the defender leads to an information set I for the attacker. Different defender's actions in different networks can lead to the same information set $I \in \tau$. Then, in every information set I, the attacker chooses a best response action to maximize his expected utility.

$$max \sum_{x \epsilon X} \sum_{j \epsilon D} \sum_{i \epsilon K} U_d(x, j, i)\ d_j^x P^x a_i^{\chi(x,j)} \tag{1}$$

$$s.t. \sum_{(x,j):\chi(x,j)=I} U_a(x, j, i) d_j^x P^x\ a_i^I \geq \sum_{(x,j):\chi(x,j)=I} U_a(x, j, i')\ d_j^x P^x\ a_i^I$$

$$\forall i, i' \in K \quad \forall I \in \tau \tag{2}$$

$$d_j^x \geq 0 \quad \forall x \in X \quad \forall j \in D \tag{3}$$

$$\sum_{j \epsilon D} d_j^x = 1 \quad \forall x \in X \tag{4}$$

$$\sum_{i \epsilon K} a_i^I = 1 \quad \forall I \in \tau \tag{5}$$

The program's objective is to maximize the defender's expected utility, assuming that the attacker will also play a best response. In the above program, the only unknown variables are the defender's actions D (the strategies of a defender in a network $x \in X$) and the attacker's actions a^I. The inequality in Equation 2 ensures that the attacker plays his best response in this game, setting the binary variable a_i^I to 1 only for the best response i in each information set. Equation 3 ensures that the defender strategies in a network x is a valid probability distribution. Equation 4 makes sure that all probability for all network configurations sum to 1. Finally, Equation 5 ensures that the attacker plays pure strategies.

4. Empirical Study of HFSG

The HFSG game model allows us to study the strategic aspects of cyber deception against a sophisticated adversary who may be able to detect the deception using additional observations and analysis. In particular, we can evaluate the effectiveness of cyber deception under several different realistic assumptions about the costs and benefits of deception, as well as

the abilities of the players. We identify cases where deception is highly beneficial, as well as some cases where deception has limited or no value. We also show that in some cases, using two-sided deception is critical to the effectiveness of deception methods.

4.1. Measuring the Similarity of Features

One of the key components of our model is that real hosts and honeypots generate observable features according to different probability distributions. The similarity of these distributions has a large effect on the strategies in the game, and the outcome of the game. Intuitively, if out-of-the-box honeypot solutions look indistinguishable from existing nodes on the network the deception will be effective without any additional intervention by the defender. However, when the distributions of features are very dissimilar the defender should pay higher costs to modify the features to disguise the honeypots. In some cases this may not be possible, and the attacker will always be able to distinguish the real hosts and honeypots.

Measuring the similarity of the feature distributions is a somewhat subtle issue, since the defender can make changes to a limited number of features. Standard approaches such as Manhattan distance or Euclidean distance do not provide a good way to compare the similarity due to these constraints. We use a measure based on the Earth Mover's Distance (EMD) [29], which can be seen as the minimum distance required to shift one pile of earth (probability distribution) to look like another. This measure can be constrained by the legal moves, so probability is only shifted between configurations that are reachable by the defender's ability to change features.

In the experiments, we allow the defender to modify only a single feature in the network and the EMD determines the minimum cost needed to transform a weighted set of features to another where the probability of each feature configuration is the weight. The ground dissimilarity between two distributions is calculated by the Hamming distance. This distance between two distributions of equal length is the number of positions at which the comparing features are dissimilar. In other words, it measures the minimum number of feature modification or unit change required to make two sets of feature indistinguishable. We model the distance from moving the probability of one configuration (e.g., turning $[0, 0]$ into $[0, 1]$) to another by flipping of a single bit at a time with a unit cost of 1. This can be seen visually in Figure 2 where we calculate the EMD of moving the honeypot's initial distribution into that of the real node's initial distribution.

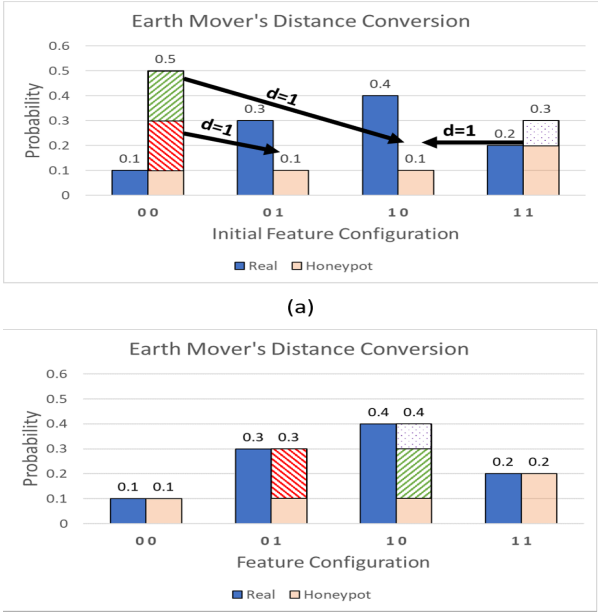


Figure 2. Earth Mover's Distance process. a) Displays the initial feature configuration probability distributions P_r and P_h and where to move slices of the distribution from P_h and b) Shows the updated P_h after the conversion, resulting in a final EMD of 0.5.

In our experiments we will often show the impact of varying levels of similarity in the feature distributions. We generated 1000 different initial distributions for the features using uniform random sampling. We then calculated the similarities using the constrained EMD and selected 100 distributions so that we have 10 distributions in each similarity interval. We randomly select these 10 for each interval from the ones that meet this similarity constraint in the original sample. This is necessary to balance the sample because random sampling produces many more distributions that are very similar than distributions that are further apart, and we need to ensure a sufficient sample size for different levels of similarity. we present the results by aggregating over the similarity intervals of 0.1 and average ten results in each interval.

4.2. Deception with Symmetric Costs

Our first experiment investigates the impact of varying the similarity of the feature distributions. We also vary the values of real host and honeypot. As the similarity of the distributions P^r and P^h decreases, we would expect a decrease in overall expected defender utility. We can see this decrease in Figures 3a and 3b as we vary the similarity measured using EMD. In

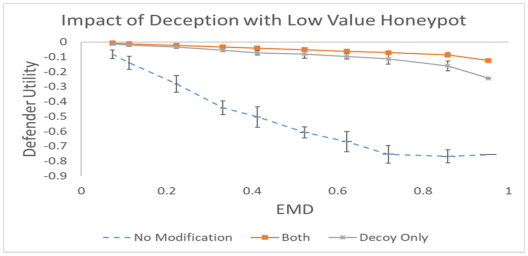
Figure	RIV	RMC		HpMC		HpIV
		F 1	F 2	F 1	F 2	
3a	1.0	0.25	0.1	0.1	0.25	0.5
3b	1.0	0.25	0.1	0.2	0.1	1.0
4 (Both (A))	1.0	0.25	0.1	0.1	0.2	0.5
4 (Both (B))	1.0	0.5	0.2	0.1	0.2	0.5
4 (Both (C))	1.0	1.0	0.5	0.1	0.2	0.5
5 (Exp-1)	1.0	0.1	∞	0.1	∞	1.0
5 (Exp-2)	1.0	0.1	∞	∞	0.1	1.0
6 (Exp-1)	1.0	0.2	0.2	0.2	0.2	1.0
6 (Exp-2)	1.0	0.15	0.25	0.25	0.15	1.0
6 (Exp-3)	1.0	0.1	0.3	0.3	0.1	1.0
6 (Exp-4)	1.0	0.05	0.35	0.35	0.05	1.0
6 (Exp-5)	1.0	0.0	0.4	0.4	0.0	1.0
8	1.0	0.25	0.1	0.2	0.1	1.0

Table 1. Parameters used in HFSG experiments. RIV denotes real system's importance value, RMC denotes real system's feature modification cost, HpIV denotes importance value of honeypot and HpMC denotes feature modification cost of honeypot. All numbers are normalized to 1

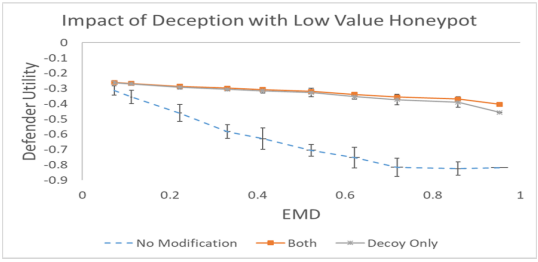
Figures 3a and 3b, we compare the utility differences between an optimal defender that can only modify the features of the honeypot (one-sided deception), an optimal defender that can modify features of *both* the honeypot and real host (two-sided deception), and a baseline defender that cannot make any modifications against a fully rational best response attacker.

In Figure 3a, the honeypot has the same importance value as the real host, while in Figure 3b, the honeypot value is half of the real host. The first observation is that in both cases the value of deception is high relative to the baseline with no deception, and this value grows dramatically as the feature distributions become more informative (higher EMD). In general, the defender does worse in cases where the hosts have different values. Two-sided deception does have a small advantage in cases with highly informative features, but the effect is small. Here, the costs of modifying the features are symmetric, so there is little advantage in being able to modify the feature on either the honeypot or the real host, since the defender can choose between these options without any penalty.

To further investigate the issue of one-sided and two-sided deception, we fix the honeypot features modification costs and increased real host modification costs as reflected in Table 1. Here, we compare how increasing the real host's feature modification negatively affects the defender's expected utility. As the cost for modifying the real hosts increases relative to the cost of modifying honeypots, the defender must make more changes on honeypots in order to maximize his utility.



(a)



(b)

Figure 3. Comparison of defender utility when the real host's importance value a) doubles that of the honeypot and b) equals that of the honeypot. Here we see one-sided deception provides a comparable defense despite a high initial dissimilarity.

Altering the real system in this case is not feasible and does not provide a good return on investment.

Traditionally network administrators avoid altering features in their real hosts on the network and simply employ one-sided deception, attempting to alter the honeypot to look like a real host. In the case where modifying a real host to look *less believable* might be be too costly or even impossible, one-sided deception is an obvious choice as demonstrated in Figure 4. However, when these real feature modifications are not too costly, we see that two-sided provides a noticeable increase in defenses when the feature distributions are increasingly dissimilar.

4.3. Deception with Asymmetric Costs

While the results so far have suggested that one-sided deception may be nearly as effective as two-sided deception, they have all focused on settings where the costs of modifying features are *symmetric* for real and fake hosts. We now investigate what happens when the costs of modifying different features are asymmetric. We start with the extreme case where some features may not be possible to modify at all.

In our examples with two features, we can set the unmodifiable features for the real and honeypot hosts to be the same or to be opposite. In Figure 5, we show the results of the game when we set the modification costs of

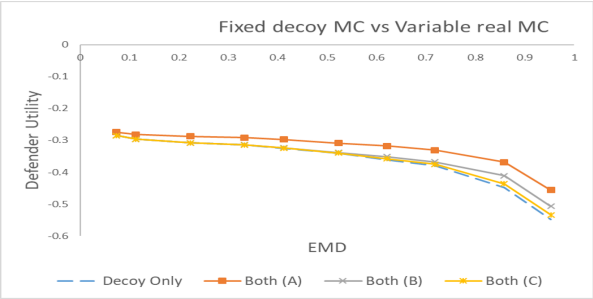


Figure 4. Comparison of defender utility when the cost of modifying the real host features is different than modifying the honeypot features.

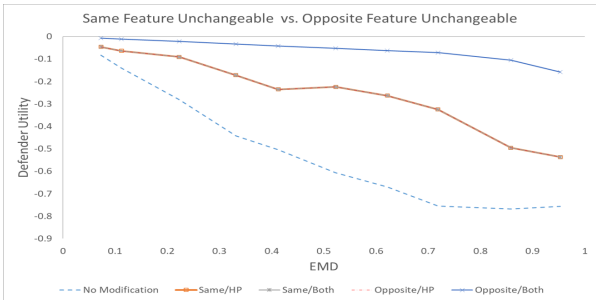


Figure 5. Comparison of defender utility when some features cannot be modified.

some features to infinity. If the same feature for the real host and honeypot are unmodifiable, then there is little the defender can do to deceive an intelligent attacker when they are highly dissimilar. However, when the features that cannot be modified are different for the real and honeypot hosts, we see a very different situation. In this case the defender benefits greatly from being able to use two-sided deception, since he can avoid the constraints by modifying either the real or fake hosts as needed.

In our next experiment, we investigate less extreme differences in the costs of modifying features. We set the costs so that they are increasingly different for real and honeypot hosts, so modifying one feature is cheap for one but expensive for the other, but not impossible. We show the results of using either one or two-sided deception for varying levels of initial feature distribution similarity in Figure 6. The specific costs are given in Table 1. We see that there is very little difference when the initial distributions are similar; this is intuitive since the attacker has little information and deception is not very valuable in these cases. However, we see a large difference when the initial distributions are informative. As the difference in the feature modification costs increases, the value of two-sided deception increases, indicating that this asymmetry is

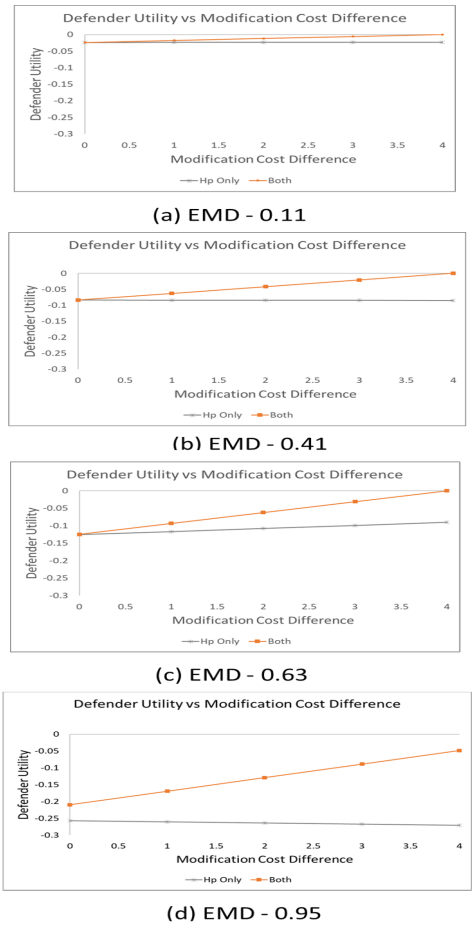


Figure 6. Impact of modification cost over various initial similarity parameters.

crucial to understanding when two-sided deception is necessary to employ effective deception tactics.

We also expect that the number of features available to the players will have a significant impact on the value of deception. While the current optimal solution algorithm does not scale well, we can evaluate the differences between small numbers of features, holding all else equal. Figure 7. presents the results of the modeling HFSG with variable number of features.We found that when the number of features is increased two-sided deception becomes more effective than one-sided deception. The defender in this case has more opportunity to alter the network by changing the features and make it the more confusing network to the attacker. However, the defender payoff decreases with more features due to the constraint on how many features he can modify and the total cost of modifying these features.

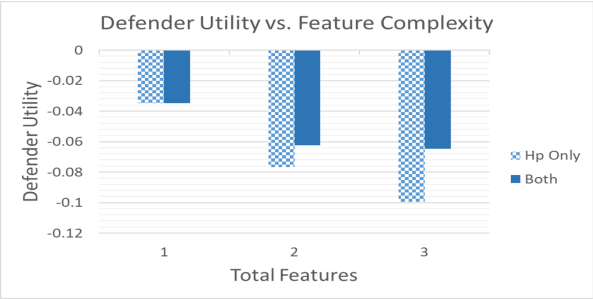


Figure 7. Comparison of defender utility when increasing the number of features.

4.4. Deception with Naïve Attackers

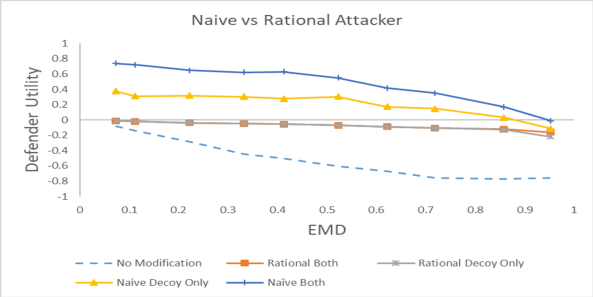


Figure 8. Comparison of defender utility of a naïve attacker versus a fully rational attacker. Here, the naïve attacker does not consider the defender's utility or strategy at all.

The previous empirical results all assumed a cautiously rational attacker who actively avoided attacking honeypots. This is a common practice, because fully rational actors present the highest threat. In cybersecurity, these fully rational attackers might be an experienced hacker or APT. However, these are not the only threats faced in cybersecurity and we cannot assume that these attacking agents are always cautious and stealthy. For example, many attacks on networks may be conducted by worms or automated scripts that are much simpler and may be much more easily fooled by deceptive strategies.

We now consider a more naïve attacker that does not consider the defender's deception. He observes the hosts on the network and assumes no modifications were made. Based on all observations for a particular network he calculates his best response, but does predict the defender's optimal strategy. The results of the experiment are shown in Figure 8 and the costs given in Table 1.

The best case is when the defender can perform two-sided deception against a naïve attacker and the worst case is when the defender performs no deceptive actions against a fully rational attacker. These two cases form an upper- and lower-bound as seen in Figure 8. Two-sided deception is more effective in this case when the feature distributions are similar, while the opposite was true for a rational attacker. Overall, deception strategies are much more effective against naïve attackers.

5. Discussion and Further Applications

Our model gives a new and more nuanced way to think about the quality of different deception strategies, and how robust they are to an adversary being able to see through the deception. We can identify which features the defender should focus on modifying to make the deception more effective, including features of the real objects. In addition, we can correctly identify cases where deception is not the best solution because the costs of creating a believable deception may be higher than the value they create. We conclude by discussing some connections to adversarial machine learning and an additional case where our model could be applied beyond honeypots.

5.1. Adversarial Learning

Recently, adversarial machine learning models have shown great promise in generating deceptive objects, focusing mostly on images and video applications [30, 31, 32], though they have the potential to generalize to many other types of deceptive objects. The most well-known approach is Generative Adversarial Networks (GAN) [33], which rely on a pair of neural networks, one to generate deceptive inputs, and the other to detect differences between real and fake inputs. The intuition for these is often that the networks are playing a zero-sum game, though the interpretation is vague and there is no formal game presented. Our model can be viewed as a formalization of the game these types of machine learning algorithms are playing, though there are some differences. We specifically consider the costs of modifying different features of the objects, as well as the possibility of modifying the real distribution in addition to the fake one. On the other hand, GANs typically are used in much larger problems with vast numbers of complex features, and they do not find optimal solutions. Also, they use abstracted representations of the feature space in the learning process, and it is not clear exactly how this works or what the implications are.

We believe that further developing and scaling this model to address more complex feature deception problems will help to understand the theoretical qualities

of GANs and related methods better. In particular, we can better understand the limits that these AML methods may have based on the costs and infeasibility of modifying features in some cases, as well as giving optimal or bounded approximations of the solutions to small feature deception games, which can then be used to provide clear quality comparisons for machine learning methods that may scale to much more complex problems but without specific quality guarantees.

5.2. Disguising Network Traffic

While we presented our model using honeypots as a motivating domain, there are many other possible applications. We briefly discuss another example here to make this point. There are many reasons to disguise network traffic to look like other traffic; defenders may wish to do this to generate fake traffic to support honeypots or to conceal the properties of real traffic on their networks otherwise. Attackers also may want to make their network traffic appear similar to real traffic to avoid detection.

While network traffic, in general, has a very large number of possible features, an increasing fraction of traffic is encrypted, which hides many of the deep features of the data. However, it is still possible to do an analysis of encrypted traffic based on the source, destination, routing, timing and quantity characteristics, etc. Our model can be used to analyze how to optimize the properties of real and decoy traffic to improve the effectiveness of the decoy traffic, based on the costs of modifying different features. For example, modifying traffic to be sent more frequently will clearly have costs in increased network congestion, while modifying some features of the real traffic may not be feasible at all (e.g., the source and destination). Even simple versions of our model with relatively few features could be used to optimize decoy network traffic in encrypted settings, where there is limited observable information about the traffic. The unencrypted case allows for many more possible features, so it would require larger and more complex versions of our model to analyze, which would require more scalable algorithms to solve exactly using our model, or the application of approximation methods and adversarial learning techniques.

5.3. Limitations

The time and memory complexities of the game model depend on n, k, feature modification options, and the amount of sampling; which makes the model grow exponentially. To avoid computational complexity, we tested our model with two machines, one each of type (*real* and *honeypot*) with two features in each. Extending the model to include more machine types and features is straightforward, although the optimization problem will become much more difficult to solve. A scalable algorithm will need to be developed to solve larger size games.

6. Conclusions

Deception is becoming an increasingly crucial tool for both attackers and defenders in cybersecurity domains. However, existing formal models provide little guidance on the effectiveness of deception, the amount of effort needed to sufficiently disguise deceptive objects against motivated attackers, of the limits of deception based on the costs of modifying the features of the deceptive objects. Also, most analyses only consider how to make deceptive objects look real, and not how real objects can be modified to look more like deceptive ones to make the task of deception easier. We present a formal game-theoretic model of this problem, capturing the key problem of disguising deceptive objects among real objects when an attacker may observe external features/characteristics.

Our model of HFSG allows us to investigate many aspects of how a defender should optimize efforts to conceal deceptive objects, which can be applied to honeypots, disguising network traffic, and other domains. This also gives a more theoretical foundation to understand the benefits and limitations of adversarial learning methods for generating deceptive objects. We show that the symmetry or asymmetry of the costs of modifying features is critical to whether we need to consider 2-sided deception as part of the strategy, and we also show that in some cases deception is either unnecessary or too costly to be effective. Also, the sophistication of the attackers makes a great difference; in cases with naïve attackers deception is even more effective, even when considering a low-cost strategy.

7. Acknowledgement

The Army Research Office supported this work under award W911NF-17-1-0370.

References

[1] M. I. Center, "Apt1: Exposing one of china's cyber espionage units," *Mandiant, Tech. Rep*, 2013. https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf.

[2] L. Spitzner, *Honeypots: tracking hackers*, vol. 1. Addison-Wesley Boston, 2002.

[3] C. Kiekintveld, V. Lisy, and R. Pibil, "Game-theoretic foundations for the strategic use of honeypots in network

security," *Advances in Information Security*, vol. 56, pp. 81–101, 2015.

[4] S. Achleitner, T. La Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha, "Cyber deception: Virtual networks to defend insider reconnaissance," in *Proceedings of the 8th ACM CCS international workshop on managing insider security threats*, pp. 57–68, ACM, 2016.

[5] N. Virvilis, B. Vanautgaerden, and O. S. Serrano, "Changing the game: The art of deceiving sophisticated attackers," in *2014 6th International Conference On Cyber Conflict (CyCon 2014)*, pp. 87–97, IEEE, 2014.

[6] C. C. Zou and R. Cunningham, "Honeypot-aware advanced botnet construction and maintenance," in *International Conference on Dependable Systems and Networks (DSN'06)*, pp. 199–208, IEEE, 2006.

[7] N. Krawetz, "Anti-honeypot technology," *IEEE Security & Privacy*, vol. 2, no. 1, pp. 76–79, 2004.

[8] C. Stoll, *The cuckoo's egg: tracking a spy through the maze of computer espionage*. Doubleday, 1989.

[9] A. Mairh, D. Barik, K. Verma, and D. Jena, "Honeypot in network security: a survey," in *Proceedings of the 2011 international conference on communication, computing & security*, pp. 600–605, ACM, 2011.

[10] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A survey on honeypot software and data analysis," *arXiv preprint arXiv:1608.06249*, 2016.

[11] M. L. Bringer, C. A. Chelmecki, and H. Fujinoki, "A survey: Recent advances and future trends in honeypot research," *International Journal of Computer Network and Information Security*, vol. 4, no. 10, p. 63, 2012.

[12] N. Provos, "Honeyd-a virtual honeypot daemon," in *10th DFN-CERT Workshop, Hamburg, Germany*, vol. 2, p. 4, 2003.

[13] N. Garg and D. Grosu, "Deception in honeynets: A game-theoretic analysis," in *2007 IEEE SMC Information Assurance and Security Workshop*, pp. 107–113, IEEE, 2007.

[14] R. Píbil, V. Lisỳ, C. Kiekintveld, B. Bošanskỳ, and M. Pěchouček, "Game theoretic model of strategic honeypot selection in computer networks," in *International Conference on Decision and Game Theory for Security*, pp. 201–220, Springer, 2012.

[15] T. Alpcan and T. Başar, *Network security: A decision and game-theoretic approach*. Cambridge University Press, 2010.

[16] A. Laszka, Y. Vorobeychik, and X. D. Koutsoukos, "Optimal personalized filtering against spear-phishing attacks.," in *AAAI*, 2015.

[17] E. Serra, S. Jajodia, A. Pugliese, A. Rullo, and V. Subrahmanian, "Pareto-optimal adversarial defense of enterprise systems," *ACM Transactions on Information and System Security (TISSEC)*, vol. 17, no. 3, p. 11, 2015.

[18] A. Schlenker, H. Xu, M. Guirguis, C. Kiekintveld, A. Sinha, M. Tambe, S. Sonya, D. Balderas, and N. Dunstatter, "Don't bury your head in warnings: A game-theoretic approach for intelligent allocation of cyber-security alerts," in *IJCAI*, 2017.

[19] A. Schlenker, O. Thakoor, H. Xu, F. Fang, M. Tambe, L. Tran-Thanh, P. Vayanos, and Y. Vorobeychik, "Deceiving cyber adversaries: A game theoretic approach," in *AAMAS*, 2018.

[20] W. Wang and B. Zeng, "A two-stage deception game for network defense," in *Decision and Game Theory for Security*, 2018.

[21] K. Wang, M. Du, S. Maharjan, and Y. Sun, "Strategic honeypot game model for distributed denial of service attacks in the smart grid," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2474–2482, 2017.

[22] Q. D. La, T. Q. Quek, J. Lee, S. Jin, and H. Zhu, "Deceptive attack and defense game in honeypot-enabled networks for the internet of things," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1025–1035, 2016.

[23] M. Du, Y. Li, Q. Lu, and K. Wang, "Bayesian game based pseudo honeypot model in social networks," in *International Conference on Cloud Computing and Security*, pp. 62–71, Springer, 2017.

[24] N. C. Rowe, E. J. Custy, and B. T. Duong, "Defending cyberspace with fake honeypots," *JOURNAL OF COMPUTERS*, vol. 2, no. 2, p. 25, 2007.

[25] T. E. Carroll and D. Grosu, "A game theoretic investigation of deception in network security," *Security and Communication Networks*, vol. 4, no. 10, pp. 1162–1172, 2011.

[26] L. Shi, J. Zhao, L. Jiang, W. Xing, J. Gong, and X. Liu, "Game theoretic simulation on the mimicry honeypot," *Wuhan University Journal of Natural Sciences*, vol. 21, no. 1, pp. 69–74, 2016.

[27] F. De Gaspari, S. Jajodia, L. V. Mancini, and A. Panico, "Ahead: A new architecture for active defense," in *Proceedings of the 2016 ACM Workshop on Automated Decision Making for Active Cyber Defense*, pp. 11–16, ACM, 2016.

[28] C. Kroer and T. Sandholm, "Extensive-form game abstraction with bounds," in *Proceedings of the fifteenth ACM conference on Economics and computation*, pp. 621–638, ACM, 2014.

[29] G. Monge, "Mémoire sur la théorie des déblais et des remblais," *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

[30] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," *arXiv preprint arXiv:1606.04435*, 2016.

[31] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pp. 43–58, ACM, 2011.

[32] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[33] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.